PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-263262
(43)Date of publication of application : 11.10.1996

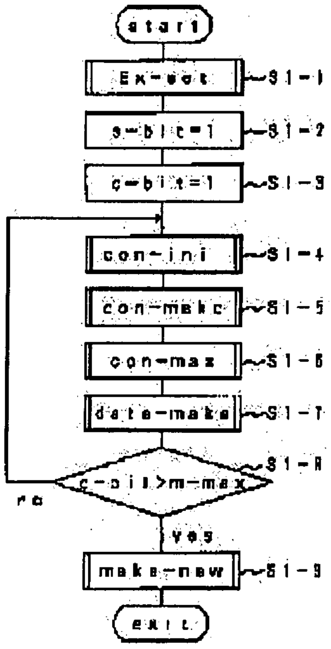
(51)Int.Cl. G06F 5/00
G06T 9/00
H03M 7/46
H04N 1/413

(21)Application number : 07-060900
(22)Date of filing : 20.03.1995

(71)Applicant : OKI DATA:KK
(72)Inventor : TAKAHASHI YOSHINORI

(54) COMPRESSION METHOD FOR PRINTING DATA

(57)Abstract:
PURPOSE: To provide a compression method by which compression can efficiently be executed even if data on a line which is to be converted and data on an immediately preceding line are hardly matched with each other.
CONSTITUTION: An exclusive OR Ex (n, m) is obtd. between data on the line to be converted and data on the same bit position before n-lines (step 1-1). The head bit position variable s-bit which is not converted yet and the variable c-bit of a pointer showing a bit position, under retrieval are initialized (steps 1-2 and 1-3) and the number of continuous identical bits con(n) is initialized (step 1-4). Then, a sub-routine con-make is called, and the number of the continuous identical bits in respective lines preceding to the head bit position which is not converted yet is retrieved. Then, the result is stored (step 1-5) in con(n). The largest con (n) among generated pieces of con(n) is searched (step 1--6). The sub-routine data-make is called and converted data is generated(step 1-7) in accordance with the maximum number of continuous identical bits obtained as a result.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision

of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]
[Claim 1] It searches whether it is in two or more lines which the print data of the line concerned which change, and the same data precede with the line concerned in the approach of compressing print data. The number of the same data which continue in each of said multi-line to precede is computed. The compression approach of the print data characterized by changing the data of said line concerned by data similarly [said maintenance line] when the same number of data which a data-hold line and said line concerned follow like the longest continuation with most same data which continues among said multi-lines to precede is the same more than as a predetermined number.
[Claim 2] Conversion of the data of said line concerned in data is the compression approach of the print data according to claim 1 performed by the bit which shows that it changes by data similarly [this maintenance line], the bit which shows the number of bits like continuation, and the bit which shows the precedence line count of this maintenance line similarly [a data-hold line] like said longest continuation.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application] This invention relates to the approach of compressing, in case especially print data are kept about the approach of compressing the print data which print by the printer.
[0002]
[Description of the Prior Art] The explanatory view in which drawing 2 shows a printing result, and drawing 3 are the explanatory views showing the print data for obtaining the printing result of drawing 2 . Drawing 2 is a thing expressing the assembly of the minimum physics size prime spot (dot) which can be printed, and drawing 3 is expressing the dot as an assembly of binary value (the information on 1 or 0, bit).
[0003] In drawing 3 , 1 to the 12th and the combination of 0 are not changing from the left after the 6th line. The data of each line are compared with the data of the direct continued line. Then, the part of the same combination of a bit as the direct continued line "the combination of m bits which continues from here The part from which it changes into the information are the same as the direct previous line", and the combination of the direct continued line and a bit differs The last twist to change can also generate a refreshable bit string with the small number of bits by storing as information "n bits of continuing n-bit information which continues from here are allotted." By such compression approach, a bit to show whether a part for that it is that the information as the direct previous line that the continuing data are the same is shown, or what bit is the same as the direct previous line is needed.
[0004]
[Problem(s) to be Solved by the Invention] However, by the above-mentioned conventional compression approach, when printing to a dot as shown in drawing 4 , data cannot be compressed. Drawing 5 is a binary-value table corresponding to the dot of drawing 4 . The data of each line are hardly in agreement with the data of the direct previous line so that drawing 5 may show. Therefore, when it compresses by the above-mentioned conventional approach, the part and the number of bits for which a bit to show whether a part for that it is that the information as the direct previous line that the continuing data are the same is shown, or what bit is the same as the direct previous line is needed will increase.
[0005]
[Means for Solving the Problem] In order to solve the above-mentioned technical problem the compression approach of this invention It searches whether it is in two or more lines which the print data of the line concerned which change, and the same data precede with the line concerned. The number of the same data which continue in each of said multi-line to precede is computed. When the same number of data which a data-hold line and said line concerned follow like the longest continuation with most same data which continues among said multi-lines to precede is the same more than as a predetermined number, the data of said line concerned are changed by data similarly [said maintenance line].
[0006]
[Function] Since according to this invention which has the above-mentioned configuration the print data of the line concerned which change, and the same data compare whether it is in two

or more lines preceded with the line concerned and specify and compress the print data of the line concerned, and the data of the precedence line which has the same continuous data from the inside, it is efficiently compressible.
[0007]
[Example] Hereafter, the example concerning this invention is explained according to a drawing. In addition, the same sign is given to the element common to each drawing. Drawing 1 is a flow chart which shows actuation of the example concerning this invention.
[0008] First, although the dot combination of the 3rd line has the 1st train, two trains, the odd number train of 5-29, 30 trains, and 32 the same trains as compared with the 2nd last line, as compared with the 1st line, the bit string to the 4th - the 32nd train is completely the same in drawing 5 . Thus, by extending the range of comparative to the multi-line preceded with an applicable line Even if it changes into the information "the next n bits are the same as the data of the same bit position of the m line continued line" and compares to a front multi-line, when the combination of the bit same in addition as an applicable line does not exist By storing as information "n bits of continuing n-bit information which continues from here are allotted", the last twist to change can also generate a refreshable bit string with the small number of bits.
[0009] In order to make the two above-mentioned kinds of information intermingled, in this example, the format shown in drawing 6 and drawing 7 is used. Drawing 6 shows format ** and drawing 7 shows format **. In this example, the precedence line count which makes the maximum number of bits specified in one format the object of 16 and a comparison is set to a maximum of 8. The most efficient value changes with combination of the actual dot train from which these two numeric values are set as the object of conversion.
[0010] Since the continuing data express with this example whether it is that the same information as a precedence line is shown, 1 bit is allotted. In format **601 shown in drawing 6 , it is shown that "1" is the same information as a precedence line, and it is shown that "0" is the combination of a new bit. For example, in drawing 6 , since the format discernment bit 602 is "1", the continuing data are the same information as a precedence line. The numeric value +1 in which the number of bits 603 is stored shows [the amount of what bit] continuously whether it is the same combination as a precedence line like continuation. Moreover, the precedence line count 604 shows whether it is the same as that of how many line before continuously for the numeric value +1 stored. Therefore, as shown in drawing, when the number of bits 603 is [the precedence line count 604] 3 in 5 like continuation, 6 bit data show the four-line continued line and that it is the same information continuously.
[0011] In drawing 7 , the format discernment bit 702 is "0" in format **701. The insertion number of bits 703 shows a part for what bit is continuously inserted for the numeric value +1 stored. The insertion data 704 show the data itself which should be inserted. That is, as shown in drawing 7 , when the insertion number of bits 703 is 4, it is shown that 5 bit data 704 are newly inserted continuously.
[0012] since the data of the 1st line of the binary-value table shown in drawing 5 do not have the object to compare -- all -- format ** -- becoming -- therefore, the format discernment bit 702 -- "0" and the insertion number of bits 703 -- 15 and the insertion data 704 -- binary value -- "0111010101010101" -- the insertion data 704 are continuously set to "0101010101010111" by formula ** similarly. Consequently, data of the 1st line become as they are shown in drawing 8 .
[0013] About the 2nd line, it is everybody's which is 4 bits which the bit distributed similarly as compared with the 1st line, and all become format ** as well as the 1st line, and the 2nd line data bring a result as shown in drawing 9 .
[0014] Moreover, although 2 bits of the beginning are the same as that of the 2nd line about the 3rd line, if it expresses by format **, 8 bits will be required and the number of bits will increase from 7 bits in format **. Then, it expresses by format ** including the 3rd bit. Since it is the same as that of the 1st line, the whole of 4th bit - the 32nd bit is expressed by format **. 4th bit - the 19th bit -- "-- continuous 16 bits -- two-line before -- the same -- " -- ** -- since it becomes the contents to say -- a format discernment bit -- 602 -- "1" -- 15 and a precedence line count are similarly set to 1 by the number of bits. 20th bit - the 32nd bit -- "--

continuous 13 bits -- two-line before -- the same -- " -- ** -- since it becomes the contents 'to' say -- a format discernment bit -- 602 -- "1" -- 12 and a precedence line count are similarly set to 1 by the number of bits. Consequently, the 3rd line data become as they are shown in drawing 10 .
[0015] Continuously, when the same, it considers [what bit] whether the direction which adopted format ** rather than format ** has little need number of bits here. Format ** needs 8 bits fixed. If continuous x bits are expressed by format **, since 5 bits will be added to the x-bit data itself, a total (x+5) bit is needed. If eight or more continue [this number of bits], more than [4 or more], i.e., 4 bits, continue [x], and format ** is adopted when it is the same as that of a precedence line, the need number of bits will be the same as format **, or will become less than format **.
[0016] Drawing 11 is the block diagram of the printer which applies the example of this invention. In this drawing, CPU111 compresses the print data in this example while controlling the whole actuation to a printer. ROM113, RAM114, and a port 115 are connected to CPU111 through the bus 112. A control program etc. is stored in ROM113 and print data are temporarily stored in RAM114. The high order interface circuitry 116 and the print station section drive circuit 117 are connected to a port 115, and the print station section 118 is further connected to the print station section drive circuit 117. The high order equipment which is not illustrated is connected to the high order interface circuitry 116, and print data are sent to a printer from this high order equipment.
[0017] Next, compression actuation of print data is explained according to each flow chart. The flow chart shown in drawing 1 is a main routine which shows the whole algorithm of this example, and calls a subroutine in some steps. Here, the algorithm used by each subroutine is explained.
[0018] First, cur (m) expresses the m-th bit of the line concerned, and Bit (n, m) expresses the m-th bit of n lines [of the lines concerned] ago. That is, it has left for the premise that the bit data which correspond beforehand are stored in cur (m) and Bit (n, m) here. Ex (n, m) It is the exclusive OR (notation # is the operator of an exclusive OR) of the data of the same bit position of a line and n concerned lines ago, and is Ex (n, m). When it is 0, it can be said that the same data as the m-th bit of the line concerned are n-line ago. The number of combination of the bit as the line concerned which exists continuously from the bit position under retrieval n-line ago with same con (n) is stored. s-bit It is conversion of the unconverted head bit position, and is c-bit. It is the variable of the pointer in which the bit position is shown during retrieval. m-max The total bit width of face of the line concerned is expressed. In addition, since the format to be used is made into format ** and format ** which are shown in drawing 6 and drawing 7 , the precedence line count 604 which searches a bit similarly is 8, and the maximum of the number of bits 603 and the insertion number of bits 703 is setting it to 16 like continuation. Moreover, exception handling in case the number of the lines preceded with the line concerned is less than eight is not included in actuation explained below.
[0019] In the flow chart of drawing 1 , subroutine Ex-set is called first, and it is Ex (n, m). It generates. Drawing 12 is a flow chart which shows subroutine Ex-set. In drawing 12 , the variable i with which the number of a precedence line is expressed first is initialized to 1 (step 12-1), and the variable j with which the bit position is expressed below is initialized to 1 (step 12-2). step 12-3 -- exclusive OR Ex (i, j) of the data of the bit position of the beginning of a line and one concerned line ago asking -- step 12-4 -- the bit position -- changing -- up to the data of the last bit position -- exclusive OR Ex (i, j) of the data of each bit position of (step 12-5), the line concerned, and one line ago It asks. Exclusive OR Ex (i, j) of the data of each bit position of a line and one concerned line ago If it asks, it is the exclusive OR Ex (i, j) of the data of each bit position of a line and two concerned lines ago next. It asks (step 12-6). The above processing is performed about the data of each bit position of eight lines ago (step 12-7).
[0020] if subroutine Ex-set is completed in drawing 1 -- unconverted head bit-position variable s-bit initializing (step 1-2) -- pointer-variable c-bit which shows the bit position during retrieval It initializes (step 1-3). Next, subroutine con-ini is called (step 1-4).
[0021] Drawing 13 is subroutine con-ini. It is the shown flow chart. In this drawing, the variable i of the number of a precedence line is first initialized to 1 (step 13-1), the number of bits con (i)

is initialized like continuation, and it is referred to as 0 (step 13-2). It carries out by eight lines which precede the above processing (step 13-3, 13-4). Thereby, 0 is altogether stored as the number of bits con (i) like continuation. Termination of this processing calls subroutine con-make next (step 1-5).
[0022] Drawing 14 is a flow chart which shows subroutine con-make. It is the bit-position variable j first used within this subroutine in this drawing Unconverted head bit-position variable s-bit It initializes (step 14-1). Thereby within this subroutine, it is s-bit. The existence of a bit etc. is searched more nearly similarly than a location. Next, the number i of a precedence line is initialized by 1 (step 14-2). Next, it judges whether bit-position data are the same as the data of a bit during retrieval of the line concerned during the i-th retrieval of a precedence line (step 14-3), if the same, it will branch to step 14-4, and if not the same, it will branch to step 14-11.
[0023] At step 14-4, since one bit was found similarly, the number of bits con (i) is set to 1 like the continuation under retrieval. In the i-th precedence line, it searches after that what bit is continuously the same, and stores in said con (i). Here, k is used as a variable which shows the bit position under number-of-bits retrieval like continuation. That is, since the bit position is in Variable j during retrieval, Variable k is initialized by the value of j at step 14-5, in order to investigate the bit position next, it is carried out 1 **** at step 14-6, and investigates whether it is the actually same data as the line concerned at step 14-9. When it is judged at step 14-9 that it is the same, 1 **** (i) of the numbers of bits con is carried out like continuation at step 14-10, and an unconditional branch is further carried out to step 14-6 for retrieval of the next bit position. When it is judged at step 14-9 that it is not the same as the line concerned, it escapes from a retrieval loop formation immediately, it branches to step 14-11, and shifts to processing in the following precedence line.
[0024] At step 14-7, when the number of bits exceeds the maximum 16 like continuation during retrieval, escaping from a loop formation is shown. Moreover, for Variable k, since it is a variable showing the bit position, the value at the time of retrieval is maximum bit width-of-face m-max. It is not necessary to exceed and, for step 14-8, the bit-position variable k is maximum bit width-of-face m-max. If it exceeds, escaping from a loop formation is shown. The retrieval result of the number of bits is stored in con (n) by the above like continuation in precedence each line from the unconverted head bit position. After processing of subroutine con-make is completed, it is subroutine con-max next. It is called (step 1-6).
[0025] Drawing 15 is a flow chart which shows subroutine con-make. This subroutine performs processing which looks for the greatest thing among the numbers of bits con (i) like continuation of each precedence line. In this drawing, i shows the number of the precedence line under retrieval, and p is a variable holding the precedence line number of the line which has the number of bits like the maximum continuation of the i-th precedence line retrieval time. The variable i of a precedence line number is first initialized to 1 showing the direct previous line of the line concerned (step 15-1), and the bit maintenance line number p is set to 0 like the longest continuation (step 15-2). It is by this at the retrieval initiation time, and it sets up so that a bit may not exist like any line. Next, at step 15-3, the number of bits judges whether it is 0 like continuation of the line under retrieval, in the case of 0, it branches to step 15-7, and it puts the object of retrieval into the following precedence line.
[0026] When the number of bits is not 0 like continuation of the line under retrieval, it branches to step 15-4 in order to perform the comparison with the number of bits like the known longest continuation. At step 15-4, when it judges whether there is any number of bits like the known longest continuation (the number of bits does not exist like the known longest continuation when the bit maintenance line number p is 0 like the longest continuation) and the number of bits does not exist like the known longest continuation, in order to branch to step 15-6 and to make the line under retrieval into a bit maintenance line number like the longest continuation, the line number i under retrieval is substituted for p.
[0027] Moreover, when the number of bits exists like the known longest continuation Step 15-5 compares the number of bits con (i) like continuation of the line under the number of bits con (p) and retrieval like the known longest continuation. Like continuation of the line under retrieval, like the known longest continuation of the number of bits con (i), when larger than the number of

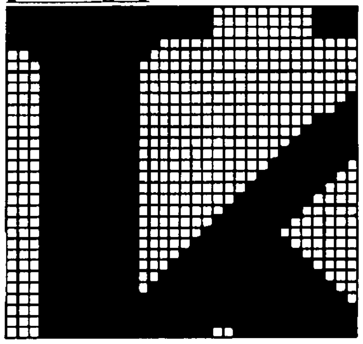
bits con (p), in order to make the line under retrieval into a bit maintenance line number like the longest continuation, the number i of the line under retrieval is substituted for step 15-6 at p. Like continuation of the line under retrieval, like the known longest continuation of the number of bits con (i), when not larger than the number of bits con (p), the bit maintenance line number p does not update, but branches to step 15-7, and puts the object of retrieval into the following precedence line like the longest continuation. Step 15-8 judges whether retrieval of the precedence line for retrieval was ended by whether the number i of the line under retrieval exceeded 8, in below the line count for retrieval, it branches to step 15-3 for retrieval of the following precedence line, and when the line count 8 for retrieval is exceeded, it ends this subroutine. The number of a bit maintenance line is stored in Variable p like the longest continuation at the time of termination of this subroutine. Variable p is 0 when it has a bit in neither of the precedence lines for retrieval similarly. Subroutine con-max After processing is completed, it is subroutine data-make next. It is called (step 1-7).
[0028] Drawing 16 is subroutine data-make. It is the shown flow chart. This subroutine data-make Subroutine con-max According to the number of bits, processing which generates translation data is performed like the obtained longest continuation. This subroutine data-make Explanation is preceded and it is this subroutine data-make. Another subroutine make-same called in inside And subroutine make-new is explained. Drawing 17 is subroutine make-same. It is the shown flow chart.
[0029] It sets to drawing 17 and is subroutine make-same. It is the subroutine which creates translation data in case there are data like [in format **] continuation in a precedence line. In addition, in this subroutine, the processing box of a parallelogram shows the processing which stores translation data in the field in memory (RAM114 shown in drawing 11). At step 17-1, "1" is first stored as a format discernment bit 602 in memory. It specifies that it is format ** by this. Next, pointer-variable c-bit which shows the bit position during retrieval at step 17-2 Unconverted head bit-position variable s-bit It stores in memory by making a difference into the number of bits 603 like continuation. At continuing step 17-3, although the precedence line count 604 of format ** is stored, since Variable p is setting the number of the direct previous line to 1, in order to make it correspond to format **, p-1 is stored.
[0030] Drawing 18 is a flow chart which shows subroutine make-new. This subroutine make-new is a subroutine which creates translation data by format **, when data do not exist like any precedence line for retrieval. Also in this subroutine, the processing box of a parallelogram shows the processing which stores translation data in the field in memory (RAM114 shown in drawing 11). First, at step 18-1, "0" is stored in memory as a format discernment bit 702, and it specifies that it is format **. Pointer-variable c-bit which shows the bit position during retrieval at step 18-2 Unconverted head bit-position variable s-bit It stores in memory by making a difference into the insertion number of bits 703. Pointer-variable c-bit which shows the bit position during retrieval It can use and (c-bit)-(second-bit) can be allotted as the insertion number of bits 703 of format **.
[0031] At step 18-6, the insertion data 704 of format ** are stored from step 18-3. It is used as a pointer of the bit position which should be stored, and Variable i is unconverted bit-position variable s-bit at step 18-3. It is substituted and initialized, the bit data cur of the line concerned (i) are stored at step 18-4, the pointer increased by every [1] at step 18-5 is followed, and a pointer is c-bit at step 18-6. It stores in memory continuously until it exceeds.
[0032] next, the flow chart of drawing 16 -- following -- subroutine data-make ******** -- it explains. When the number of bits con (p) exceeds 4 like continuation in this subroutine, translation data is stored by format **, and it is unconverted head bit-position variable s-bit. It updates. When other When the unconverted number of bits ((c-bit)- (second-bit +1)) tends to exceed the maximum bit length 16 specified by format **, translation data is stored by format **, and it is unconverted head bit-position variable s-bit. When not updating and exceeding, it is only c-bit. It is increasing one time.
[0033] First, at step 16-1, when Variable p judges by 0 or 1 (when it does not exist) (when it exists) and exists [whether a bit exists like the precedence line for retrieval, and], it branches to step 16-2, and when it does not exist, it branches to step 16-3. At step 16-2, since the

number of bits after the direction of format ** changing decreases when the number of bits is four or less like continuation, as mentioned above, when the number of bits is four or less like continuation, the same processing as the case where branch to step 16-3 and a bit does not exist similarly is performed.
[0034] When the number of bits exceeds 4 like continuation at step 16-2, it branches to step 16-4, and judges whether the bit position is equal during the unconverted head bit position and retrieval. It is here, and when it is judged that it is not equal, since data must be changed according to format **, according to format **, it is changed [during the unconverted head bit position and retrieval] from step 16-5 at step 16-8 up to 1 bit [of a bit] before. At step 16-5, in order to change the data of a 1 bit [of the bit position] before according to format ** during retrieval, a bit-position pointer variable is reduced by one during retrieval. The preparations which call subroutine make-new at step 16-6 next by this are made. At step 16-6, subroutine make-new is called, subroutine make-new processing in which it explained by drawing 18 is performed, and data are changed according to format **. c-bit amended at step 16-5 in step 16-7 c-bit which restored the contents and was restored at step 16-8 It is a value s-bit It substitutes and the unconverted head bit position is amended.
[0035] At step 16-4, when it is judged during the unconverted head bit position and retrieval that the bit position is equal, it branches to step 16-9. Subroutine make-same called at the following step 16-10 in step 16-9 Variable c-bit used by processing It is c-bit in order to double with semantics. By adding con (p) and reducing one to a pan, it is c-bit. The contents are similarly made into the bit trailing-edge location. At step 16-10, it is subroutine make-same. It calls, as drawing 17 explained, the translation data of format ** is stored, and they are s-bit and c-bit at step 16-11 and step 16-12. It updates.
[0036] pointer-variable c-bit which shows the bit position during retrieval at step 16-3 Unconverted head bit-position variable s-bit receiving -- 15 -- in many (16 bits of bits do not exist like continuation continuously), in order to change the data to a line bit during the retrieval at the time, it branches to step 16-13, subroutine make-new is called, and data are changed according to format **. And at step 16-14 and step 16-16, it is s-bit. And c-bit It updates.
[0037] It is variable c-bit at step 16-3. s-bit It is pointer-variable c-bit which branches to step 16-15 and shows the bit position during retrieval when a difference is less than 15. It judges whether the maximum bit width of face is reached, when having reached, processing is ended, when having not reached, it branches to step 16-16, and it is c-bit. It updates.
[0038] When an unconverted bit reaches the maximum bit width of face, it is a step for escaping from a loop formation, step 1-8 in drawing 1 branches to step 1-9, when an unconverted bit reaches the maximum bit width of face, and when other, it branches to step 1-4. At step 1-9, when there is an unconverted bit, subroutine make-new which changes them into format ** is called, and it processes.
[0039] The result of having changed the binary-value table shown in drawing 5 based on the conversion algorithm described above is shown in drawing 19 . When it changes by this example, 960-bit data are changed into 696 bits, and are compressed to about 72%.
[0040]
[Effect of the Invention] As explained to the detail above, according to this invention, the data of the line concerned are compared with the data of the multi-line to precede, and since the line data which have the same bit as the line concerned for a long time from the inside are specified and compressed, it is efficiently compressible so that the number of bits of print data which are hardly in agreement with the data of the direct previous line decreases.

[Translation done.]

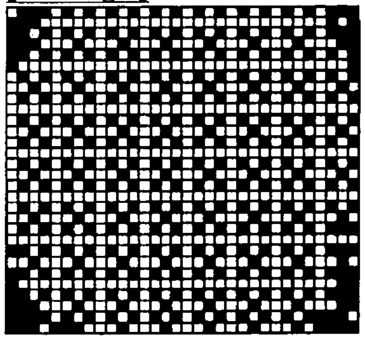
DRAWINGS

[Drawing 2]



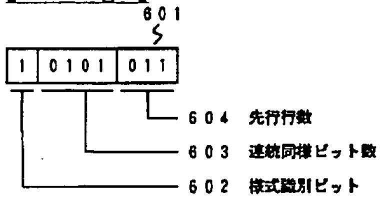
印刷結果を示す説明図

[Drawing 4]



他の印刷結果を示す説明図

[Drawing 6]



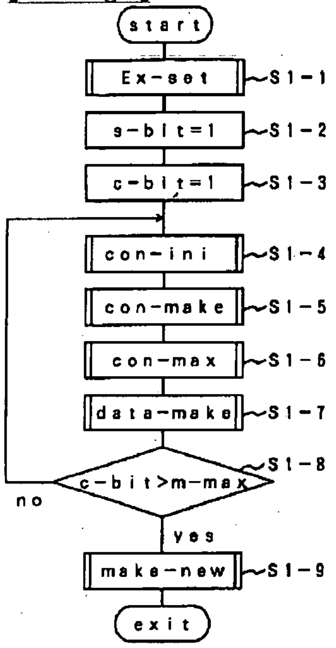
圧縮データの様式①を示す説明図

[Drawing 10]

0	0010	110	1	1111	001	1	1100	001

第３行データを示す説明図

[Drawing 1]



実施例の動作を示すフローチャート

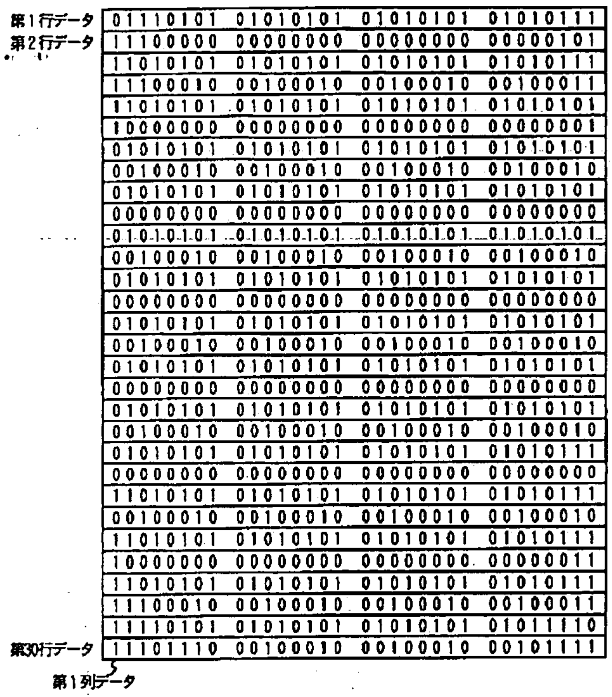
[Drawing 3]

第1行データ	11111111	11111111	11111000	00001111
第2行データ	11111111	11111111	11111000	00001111
	11111111	11111111	11111000	00001111
	11111111	11111100	00000000	00000000
	00111111	11111000	00000000	00000000
	00011111	11110000	00000000	00000000
	00011111	11110000	00000000	00000000
	00011111	11110000	00000000	00000000
	00011111	11110000	00000000	00000001
	00011111	11110000	00000000	00000011
	00011111	11110000	00000000	00000111
	00011111	11110000	00000000	00011111
	00011111	11110000	00000000	00111111
	00011111	11110000	00000000	01111111
	00011111	11110000	00000000	11111110
	00011111	11110000	00000001	11111100
	00011111	11110000	00000011	11111000
	00011111	11110000	00000111	11110000
	00011111	11110000	00001111	11100000
	00011111	11110000	00011111	11000000
	00011111	11110000	00111111	10000000
	00011111	11110000	01111111	11000000
	00011111	11110000	11111111	11100000
	00011111	11110001	11111111	11110000
	00011111	11110011	11111111	11111000
	00011111	11110111	11111111	11111100
	00011111	11111111	11111111	11111110
	00011111	11111111	11111111	11111111
	00011111	11111111	11111111	11111111
第30行データ	00011111	11111111	11100111	11111111

第1列データ

図２の印刷結果を得るための印刷データを示す説明図

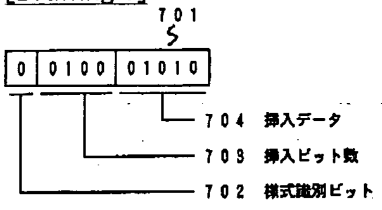
[Drawing 5]



第1行データ	01110101	01010101	01010101	01010111
第2行データ	11100000	00000000	00000000	00000101
	11010101	01010101	01010101	01010111
	11100010	00100010	00100010	00100011
	11010101	01010101	01010101	01010101
	10000000	00000000	00000000	00000001
	01010101	01010101	01010101	01010101
	00100010	00100010	00100010	00100010
	01010101	01010101	01010101	01010101
	00000000	00000000	00000000	00000000
	01010101	01010101	01010101	01010101
	00100010	00100010	00100010	00100010
	01010101	01010101	01010101	01010101
	00000000	00000000	00000000	00000000
	01010101	01010101	01010101	01010101
	00100010	00100010	00100010	00100010
	01010101	01010101	01010101	01010101
	00000000	00000000	00000000	00000000
	01010101	01010101	01010101	01010101
	00100010	00100010	00100010	00100010
	01010101	01010101	01010101	01010111
	00000000	00000000	00000000	00000000
	11010101	01010101	01010101	01010111
	00100010	00100010	00100010	00100010
	11010101	01010101	01010101	01010111
	10000000	00000000	00000000	00000011
	11010101	01010101	01010101	01010111
	11100010	00100010	00100010	00100011
	11110101	01010101	01010101	01011110
第30行データ	11101110	00100010	00100010	00101111

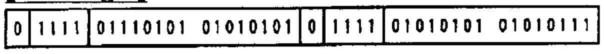
図4の印刷データを示す説明図

[Drawing 7]



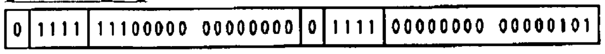
圧縮データの様式②を示す説明図

[Drawing 8]



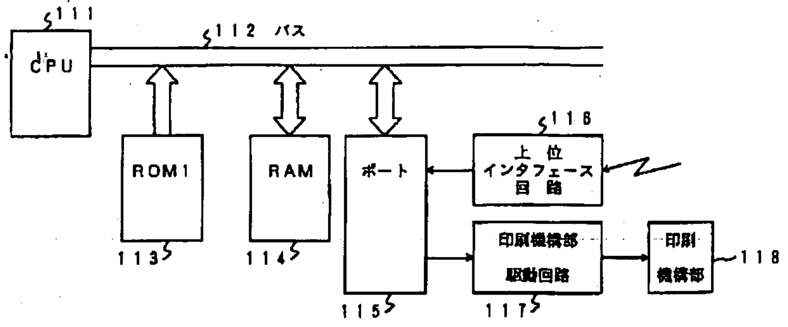
第1行データを示す説明図

[Drawing 9]



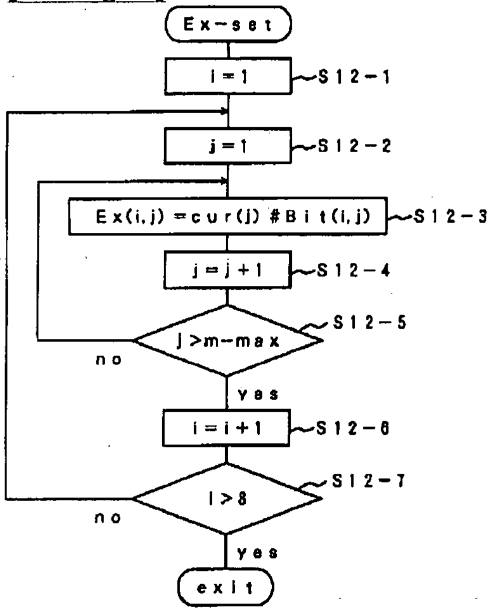
第2行データを示す説明図

[Drawing 11]



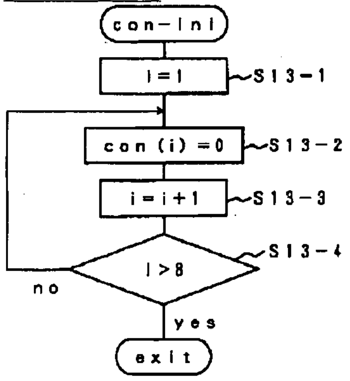
実施例を適用するプリンタのブロック図

[Drawing 12]



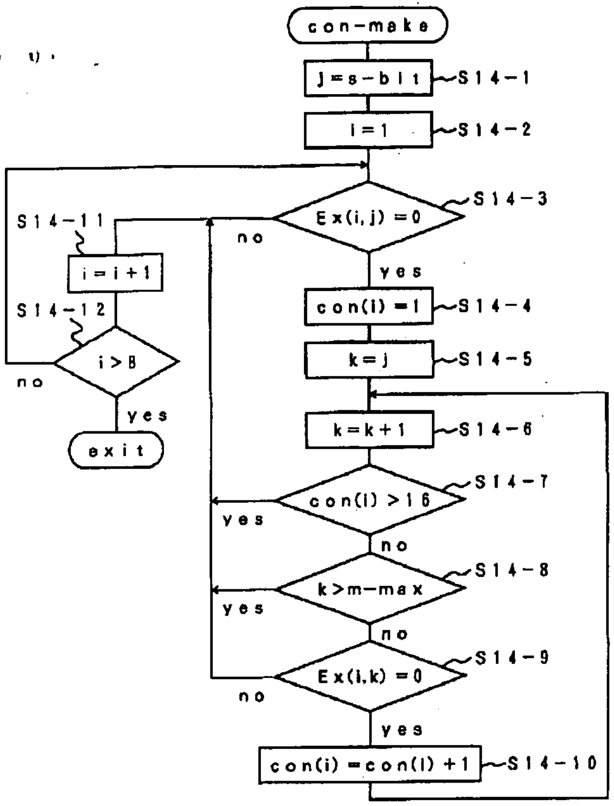
実施例のサブルーチンを示すフローチャート

[Drawing 13]



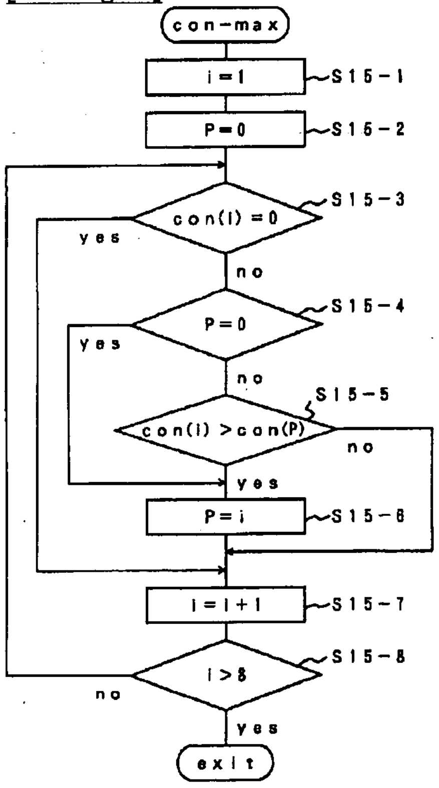
実施例のサブルーチンを示すフローチャート

[Drawing 14]



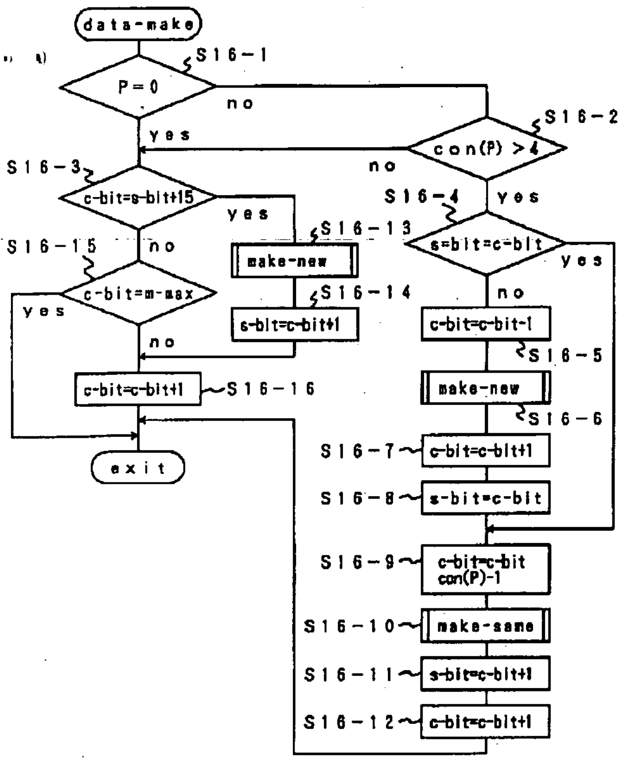
実施例のサブルーチンを示すフローチャート

[Drawing 15]



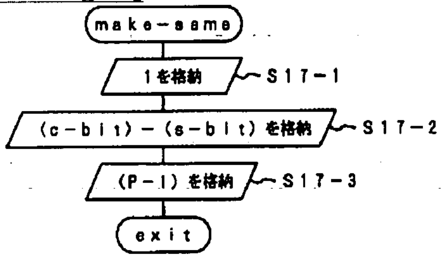
実施例のサブルーチンを示すフローチャート

[Drawing 16]



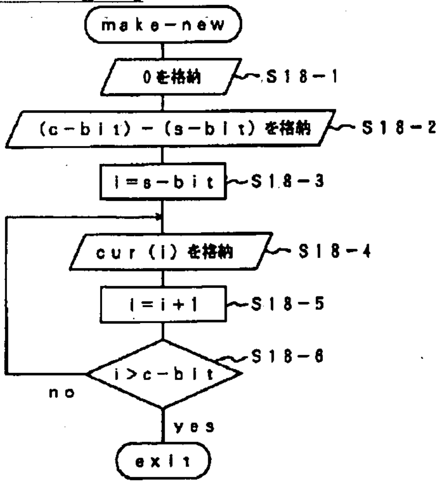
実施例のサブルーチンを示すフローチャート

[Drawing 17]



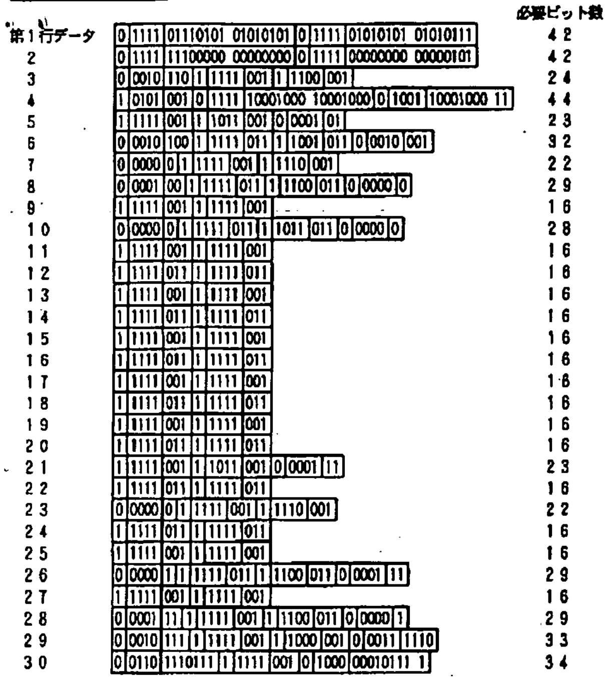
実施例のサブルーチンを示すフローチャート

[Drawing 18]



実施例のサブルーチンを示すフローチャート

[Drawing 19]



												必要ビット数
第1行データ	0	1111	01110101 01010101	0	1111	01010101 01010111						42
2	0	1111	11100000 00000000	0	1111	00000000 00000101						42
3	0	0010	110	1	1111	001	1	1100	001				24
4	1	0101	001	0	1111	10001000 10001000	0	1001	10001000 11				44
5	1	1111	001	1	1011	001	0	0001	01				23
6	0	0010	100	1	1111	011	1	1001	011	0	0010	001	32
7	0	0000	0	1	1111	001	1	1110	001				22
8	0	0001	00	1	1111	011	1	1100	011	0	0000	0	29
9	1	1111	001	1	1111	001							16
10	0	0000	0	1	1111	011	1	1011	011	0	0000	0	28
11	1	1111	001	1	1111	001							16
12	1	1111	011	1	1111	011							16
13	1	1111	001	1	1111	001							16
14	1	1111	011	1	1111	011							16
15	1	1111	001	1	1111	001							16
16	1	1111	011	1	1111	011							16
17	1	1111	001	1	1111	001							16
18	1	1111	011	1	1111	011							16
19	1	1111	001	1	1111	001							16
20	1	1111	011	1	1111	011							16
21	1	1111	001	1	1011	001	0	0001	11				23
22	1	1111	011	1	1111	011							16
23	0	0000	0	1	1111	001	1	1110	001				22
24	1	1111	011	1	1111	011							16
25	1	1111	001	1	1111	001							16
26	0	0000	1	1	1111	011	1	1100	011	0	0001	11	29
27	1	1111	001	1	1111	001							16
28	0	0001	11	1	1111	001	1	1100	011	0	0000	1	29
29	0	0010	111	1	1111	001	1	1000	001	0	0011	1110	33
30	0	0110	1110111	1	1111	001	0	1000	00010111 1				34

実施例により変換した結果を示す説明図

[Translation done.]